CPM Project

Improving the Code Release Process by Pre-Defining Testing Instructions

Clifton L. Simmons, Jr.

South Carolina Department of Motor Vehicles

January 22, 2020

Table of Contents

Table of Figures, Charts, and Tables

Problem Statement

South Carolina Department of Motor Vehicles (SCDMV) follows a Software Development Life Cycle (SCDLC) where when a software issue is encountered or an enhancement is defined, a ticket is written by a Business Analyst (BA) then assigned to a Developer. The Developer writes the code and has it promoted to the System Test[1] area. The BA then defines the test plan based on the business specifications and/or their experiences or relies on the Developer to define the acceptance test. That acceptance test is then executed in the System Test area. The testing effort may find errors that causes the code to be sent back to the Developer, and the cycle starts over. Once the ticket passes System Test, it is moved to Regression[2] and eventually Production. At any point, the ticket could be sent back to the Developer for cause. The testing and regression efforts are a predetermined length of time with no formal definition. This methodology is called Waterfall in the Information Technology (IT) industry.[3]

The Configuration Management (CM) team of the Information Technology (IT) Department at SCDMV is charged with getting onsite developed code into Production quickly and accurately. If a defect is found after the code has been moved to Production, Developers, BAs, and CM go into a heightened mode to get the issue resolved as quickly as possible. This effort can and often does delay other releases and efforts. If SCDMV can reduce the number of errors found in production, other efforts will be less impacted. Thereby, allowing this agency to better serve the citizens of South Carolina.

[1] See Glossary
[2] Ibid.
[3] See Appendix 1

Data Collection

Determining Where to Focus

Wanting to improve the SDLC, but having too many ideas on where to study, an unscientific survey was sent to a group of fifty. Business Analysts, Developers, and Configuration Management professionals were asked two questions.[4] The questions were opened ended in an effort to generate thought and to not lead them to a particular response.

An attempt was made to review the failures from MR-276. MR-276 was completed and released prior to this project being started; therefore, the Developers and BAs may have not remembered why the ticket had been failed. Of the 21 Failed Incidents (fifteen failed tickets), responses were received for only four tickets. Three responses called for better testing tools (i.e. Automated Testing) and one failed because the requirement was missed. The process of obtaining, installing, training, and executing an Automated Testing system would take approximately two years. Defining more detailed testing steps would be one of the steps needed when installing an automated testing tool. Therefore, the basis of this project will focus on better defined testing.

The Focus

Code/tickets for SCDMV are promoted by Module Release (MR). Contents of each MR are defined by the BAs with the assistance of the Developers and Managers in a review meeting. The MR is then assigned to a release date based on the expected effort and other efforts within the agency. Therefore, dates are not consistent (code is not released at specific times or intervals). However, attempts are made to promote code to Production every six to eight weeks. The manager

[4] See Appendix 13

of CM defines the release schedule at the beginning of the year; however, that schedule is often adjusted due to other efforts or the effort of the release.

That being said, the data used for this project are broken down by releases and not time periods. To give a better history or baseline of data, it was decided to gather data going back to MR-268 which was the first release of 2018.[5]

Groups of data collected for the various baselines are listed alphabetically as follows[6]:

- Failed Tickets vs Failed Incidents
- Number of Tickets Failed More Than Once
- Number of Tickets per Release
- Number of Failed Incidents
- Number of Failed Tickets
- Percent of Failed Tickets
- Percent of Times Tickets Failed
- Percent of Times Tickets Failed More Than Once

It was determined that these data groups would allow us to see if changes made impacted our outcome. While reviewing the data charts in Appendices 3 through 10, note that there are two charts. The first chart shows the data prior to and including MR-277. The second chart shows the data from MR-268 to MR-280. This was done to show the change in Mean and Sigmas.

[5] See Appendix 2
[6] See Appendices 3 – 10

Data Analysis

Determining Where to Focus

Answers from the survey of fifty[7] for the failed tickets ranged from no response received to the "Developer did not promote the correct code." Two responses that stood out were the Developer "did not have tools (printer) to test" and "did not know enough about how to test." From the response chart in Appendix 13 titled *Improvement Suggestion*, we see that the Developer and/or BA identified some form of testing to be the issue 25% of the time.

The Focus

The average number of failed incidents[8] less the average number of failed tickets[9] shows that the difference between the two averages does not gets closer than 4.1 between MR-268 and MR-277. The closest that the actual numbers get during the initial Data Analysis phase is two in MR-274.

The number of Failed Incidents, ideally, should not be greater than the number of Failed Tickets. The goal should always be no failed tickets; however, that is not a realistic expectation in IT – at some point, you will have a failed ticket.

Beginning with MR-277 when a ticket was failed by the BA, the Developer and BA were asked the following questions:

- Why was this ticket failed?
- Was there something missing in the write up?
- Was there a misunderstanding of the write up?
- Was it something else, if so what?
- Did the ticket work with the attached testing instructions? If no, did the developer follow the test instructions prior to promoting the code?

[7] See Appendix 13
[8] See Appendix 3
[9] Ibid.

Implementation Plan

Action Steps needed to complete the goal (and who performs them)

The BA who writes the ticket will work with the assigned Developer to create an acceptance test that will be added to the ticket. When the Developer is coding and has finished coding, they will execute the test plan to determine if the ticket meets the predefined conditions of the test. Likewise, the BA will execute the same test during System Test.

Timeframes and cost

Since coding for MR-278 has already begun, a reduction in failures will be difficult to determine. However, the action of adding predefined acceptance testing will begin with the tickets in MR-278 that are promoted after July 3rd. MR-279 should truly begin to show if the theory works.

Real cost will be difficult to measure since the BAs and Developers each have different rates of pay and actual time spent on the ticket is not tracked by this agency. The one measure that can be used is amount of time taken to define the acceptance test. It is estimated that predefining the acceptance test will add approximately thirty minutes per ticket.

The timeframe used will be from MR-278 to MR-280. If the theory proves correct, there will be a reduction in the number of failed tickets and incidents.

Potential obstacles and methods to overcome them

The only obstacle to overcome is “we never did it that way before” or “we tried, it did not work.” Many employees predate the current management and are set in their ways. The only real method to overcome this attitude is to have the theory prove correct. Some attitudes can be mitigated by explaining that this is a test.

Potential resources

No new or additional physical resources are needed. SCDMV currently has all resources needed in place:

- Ticketing System – Microsoft System Center Service Manager (SCSM)
- Microsoft Office
- Developers
- BAs

Communication with key stakeholders

The Applications Senior Manager and Ops Senior Manager were notified of the theory and possible solution. Their input was also solicited. The Application Manager responded, "This is what we're going to test and what the test cases are so the developer has a clear target and has testing scenarios he can use to validate his effort."

The Development Managers were notified one day before all the Business Analysts and Developers were notified. One of those managers called it "more bureaucracy."

The next level of communication was to notify the BAs and Developers of the test.[10] The initial push back from one developer stated, "Personally, I think an acceptance plan will be helpful in less than 10% of the situations, and I think it is going to cost more time than it is going to save." After explaining that approximately 42% of the failed tickets in MR-277[11] could be traced back to some form of testing effort, the Developer began to see the need for the study. A big endorsement came from the Application Senior Manager. His response ended several concerns about the perceived extra effort on behalf of the Developers.

[10] See Appendix 15
[11] See Appendix 14

Integration into standard operating procedure

Integration into the standard operating procedure is as simple as inserting the testing plan onto the *Documentation* tab of SCSM. From there the developer will find the testing steps that will be used to prove that the solution is working correctly. Tickets requiring Business Specifications will have the acceptance test defined in the appropriate section within that document. Once the Developer has completed their testing, a notation will be made in the ticket indicating that the test was successful.

Evaluation Method

The data defined in the *Data Collection* section of this document will be represented in the *Control Charts.* If the difference between the baseline (MR-268 through MR-277) analysis and the failures in the rest of the experiment are less, the experiment is successful. The greater the difference the more successful.

It is possible to see positive change above the mean; therefore, two measures were devined. Ideally, the desire is for Positive Change while Positive Shift is acceptable. Change shown on the *Control Charts* will be measured as follows:

Change	Positive Change	Negative Change
Minor	Less than the mean and greater than -1σ Or a decrease of one to five failures	Greater than the mean and less than 1σ Or an increase of one to five failures
Moderate	Less than the -1σ and greater than -2σ Or a decrease of six to ten failures	Greater than the 1σ and less than 2σ Or an increase of six to ten failures
Major	Less than the -2σ and greater than -3σ Or a decrease of eleven or more failures	Greater than the 2σ and less than 3σ Or an increase of eleven or more failures

Table 1: Measure of Change

Each ticket failure will be followed with a request for information from the BA and Developer. They will be asked the four questions defined in the *Data Analysis* section of this document. Their answers will be placed into one of six groupings and represented with a *Data Points* chart:

- *Developer did not follow test plan* – The Developer did not execute all or part of the Test Plan that was defined.
- *Developer Error* – The developer did something that caused the ticket to fail. This could be a promotion error or they inadvertently broke another section of code.
- *Misunderstood ticket* – Either the Developer or the BA misunderstood all or part of the requirement.
- *Did not Respond* – Developers and most BAs do not work in the IT Department and are not in the chain of command that would require an answer to the survey.
- *Scope Creep* – Someone, in most cases the BA, wants something added to the ticket after the code has been promoted the first time.

- *Test Plan not complete* – The Test Plan did not contain enough steps or the correct steps to ensure a successful test.

Summary and Recommendations

Barry W. Boehm in his 1981 book *Software Engineering Economics* states that the earlier a defect is found the cheaper it is to resolve. NASA confirmed this in its *Error Cost Escalation through the Project Life Cycle* study that was conducted around 2003. In that study, NASA determined that the "Cost Factor" was one to 1,615 times depending on when the defect was found in the process and the method used to resolve.[12] If we can find the errors in the development phase, the amount of production down time is greatly reduced and improving the experience of the Customer Service Representative (CSR) and the citizens of this great state.

It was realized toward the conclusion of this study that a major project at SCDMV had overlapped the *Implementation Plan* of adding test plans to the newly opened tickets. The project began with MR-274 and reached a milestone with MR-279. That project was the migration of the COBOL systems from Micro Focus Net Express COBOL to Micro Focus Visual COBOL. During that time period, dual maintenance/coding was taking place – Developers were updating both Net Express and Visual COBOL systems at the same time. Likewise, the BAs were testing in two systems. Coding in both systems ended with MR-279.

At that time, there was a decrease of four failed tickets between MR-279 and MR-280 (a decrease of four failed incidents). There was an increase of ten failed tickets between MR-273 and MR-274 (the start of the Visual COBOL project). It is possible that the Visual COBOL project contributed to some of the increase in failed tickets and incidents either by the amount of coding that had to be done or by the difference in architecture.

[12] NASA, *Error Cost Escalation through the Project Life Cycle,* https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf, (May 20, 2019).

During the first four MRs (274, 275, 276, and 277) of the Visual COBOL project, the number of failed tickets and incidents stayed fairly flat – within two tickets either way. When the creation of test plans for each ticket began with MR-278, a decrease of failed tickets was realized. The first MR (278) showed a decrease of failed tickets and incidents that was more than double than each of the previous four MRs. This was the largest decrease in failed tickets since MR-270 when there was a reduction by seventeen failed tickets and eighteen failed incidents. The number of failed tickets in MR-280 is the first time since MR-273 that there was a moderate positive change[13].

It was observed early in the study that the average of failed incidents less the average of failed tickets between MR-268 and MR-277 was 4.1 points. After implementing the predefined test plans that number was reduced to 2.67 points between MR-278 and MR-280.

One theory that has often been considered at SCDMV is the number of tickets assigned to a release is directly related to the number of ticket failures. Appendix 12 shows mixed results. MR-275 had a relatively low number of tickets in the release, but had a high number of failures. MR-275 had the highest percentage for failed tickets at 34.09%[14] and failed incidents at 43.18%.[15] The percentage of failed tickets decreased by 4.56 points between MR-277 and MR-278, by 4.04 points between MR-278 and MR-279, and 7.87 points between MR-279 and MR-280. Likewise, the percentage of failed incidents decreased by 2.08 points between MR-277 and MR-278, by 7.95 points between MR-278 and MR-279, and 8.5 points between MR-279 and MR-280.

Comparing MR-280 to releases prior to MR-273 exhibits minor to major positive change. Reviewing releases MR-276 through MR-280, we see a steady decline in failures. Based on the

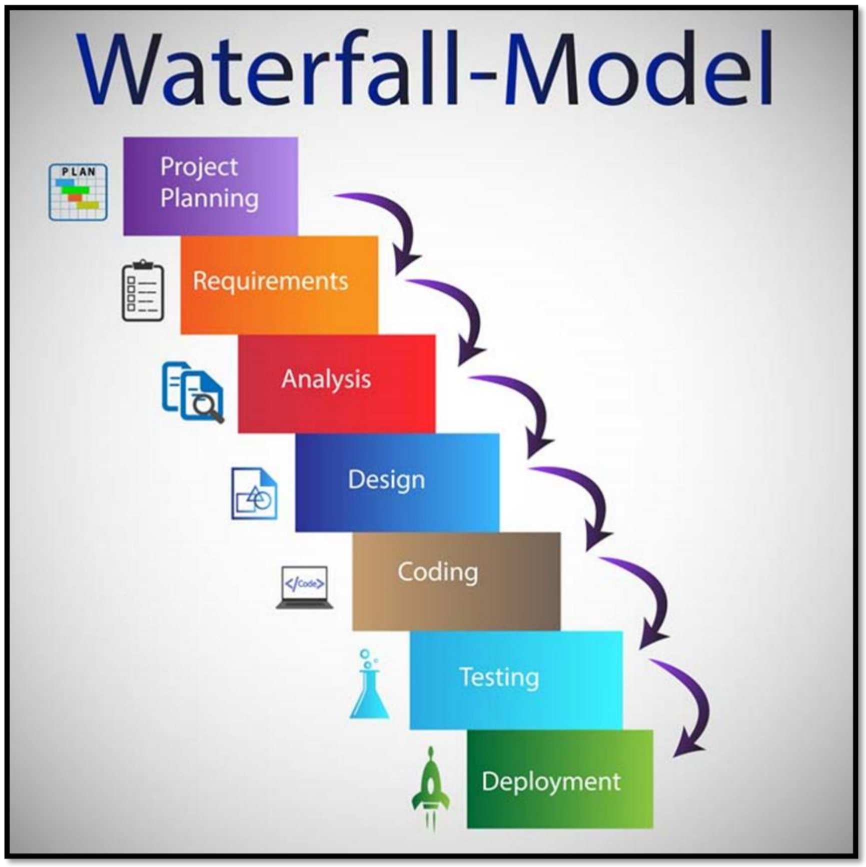
[13] See Table 1: Measure of Change
[14] See Appendix 8
[15] See Appendix 9

data above and in the appendices the implementation of test plans prior to updating code did make a positive impact and decreased the number of failed tickets and failed incidents. However, the conclusion of the Visual COBOL project could have added to that reduction. Without further study, it cannot be determined which led to the greater reduction in failures. Was it the addition of test plans or was reaching the milestone of eliminating dual maintenance?

Appendix 1: Coding Methodologies

Waterfall Methodology

First introduced by Dr. Winston W. Royce in a paper published in 1970, the waterfall model is a software development process. The waterfall model emphasizes that a logical progression of steps be taken throughout the SDLC, much like the cascading steps down an incremental waterfall.[16] The "waterfall model enforces moving to the next phase only after completion of the previous phase."[17]



[16] "Waterfall Model: What Is It and When Should You Use It?," https://airbrake.io/blog/sdlc/waterfall-model, (March 2019)
[17] "What is Waterfall Model?," Techopedia, http://www.techopedia.com/definition/14025/waterfall-model, (March 2019)

SCDMV Methodology

SCDMV's Waterfall Methodology flows as follows:

1. Problem, Enhancement, or Legislative Change is identified.
2. Ticket is written in the tracking system by a Business Analyst (BA).
3. The problem or enhancement is scheduled for release.
4. Coding
    - i) Problem
        - i. Assigned to a Developer to be resolved.
        - ii. Developer conducts development test.
    - ii) Enhancement or Legislative Change
        - i. Business Specifications are written and approved by the business area responsible for the enhancement.
        - ii. Technical Specifications are written by the development staff.
        - iii. Technical supervisors/managers review the Technical Specifications and approve or ask for revisions.
        - iv. Developer creates and/or updates the code for the enhancement.
        - v. Developer conducts development test.
5. Code is checked into a library management system after having been reviewed by a technical supervisor/manager.
6. System Testing
    - i) Configuration Management (CM) moves the code to the testing area.
    - ii) BAs conduct System Testing.
7. Regression Testing
    - i) CM moves the code to the regression area.
    - ii) BAs conduct Regression Testing.
8. Production.
    - i) CM moves the code to Production.
    - ii) BAs conduct Install Verification Plan (IVP)

Appendix 2: Module Release Schedule

Release	2018 Dates	Release	2019 Dates	Release	2020 Dates [18]
MR-268	February 16	MR-274	February 5	MR-280	January 26
MR-269	April 24	MR-276	May 14	MR-281	March 8
MR-270	July 10	MR-277	August 6	MR-282	April 28
MR-271	September 4	MR-278	September 17	MR-283	June 16
MR-272	October 23	MR-279	November 5	MR-284	August 4
MR-273	December 4			MR-285	September 22
				MR-286	November 10

Table 2 – Module Releases with Dates

[18] MR-282 through MR-286 are tentative dates

Appendix 3: Summary of Data[19]

Release	Number Failed Tickets	Number Failed Incidents	Number Tickets in Release	Number Tickets Failed More than Once	Percent Failed Tickets	Percent Failed Incidents	Percent Tickets Failed More Than Once
MR-268	12	14	61	2	19.67	22.95	3.28
MR-269	30	37	96	5	31.25	38.54	5.21
MR-270	13	19	82	4	15.85	23.17	4.88
MR-271	16	19	77	3	20.78	24.68	3.9
MR-272	11	15	41	2	26.83	36.59	4.88
MR-273	7	11	36	3	19.44	30.56	8.33
MR-274	17	19	67	1	25.37	28.36	1.49
MR-275	15	19	44	3	34.09	43.18	6.82
MR-276	15	21	65	2	23.08	32.31	3.08
MR-277	15	18	68	2	22.06	26.47	2.94
MR-278	11	15	58	3	18.97	25.86	5.17
MR-279	10	12	67	2	14.93	17.91	2.99
MR-280	4	5	74	1	5.41	6.76	1.35
Std Dev	5.96	7.02	17.64	1.04	7.08	8.91	1.89
Median	13	19	67	2	20.78	27.94	3.9
Average	13.77	17.54	65.15	2.62	21.6	27.8	4.26

Table 3: Summary of Data all MRs

[19] Calculations are rounded to two decimal places.

Release	Number Failed Tickets	Number Failed Incidents	Number Tickets in Release	Number Tickets Failed More than Once	Percent Failed Tickets	Percent Failed Incidents	Percent Tickets Failed More Than Once
MR-268	12	14	61	2	19.67	22.95	3.28
MR-269	30	37	96	5	31.25	38.54	5.21
MR-270	13	19	82	4	15.85	23.17	4.88
MR-271	16	19	77	3	20.78	24.68	3.9
MR-272	11	15	41	2	26.83	36.59	4.88
MR-273	7	11	36	3	19.44	30.56	8.33
MR-274	17	19	67	1	25.37	28.36	1.49
MR-275	15	19	44	3	34.09	43.18	6.82
MR-276	15	21	65	2	23.08	32.31	3.08
MR-277	15	18	68	2	22.06	26.47	2.94
Std Dev	6	6.93	19.04	1.16	5.59	6.83	2.01
Median	15	19	66	2.5	23.31	29.46	4.39
Average	15.2	19.3	63.7	2.7	23.99	30.83	4.48

Table 4: Summary of Data MR-268 to MR-277

Release	Number Failed Tickets	Number Failed Incidents	Number Tickets in Release	Number Tickets Failed More than Once	Percent Failed Tickets	Percent Failed Incidents	Percent Tickets Failed More Than Once
MR-278	11	15	58	3	18.97	25.86	5.17
MR-279	10	12	67	2	14.93	17.91	2.99
MR-280	4	5	74	1	5.41	6.76	1.35
Std Dev	2.65	3.51	13.75	0.58	6.06	8.23	1.48
Median	10	12	67	2	14.93	17.91	2.99
Average	9	11.67	70	2.33	13.65	17.73	3.5

Table 5: Summary of Data MR-278 to MR-280

Appendix 4: Number of Failed Tickets

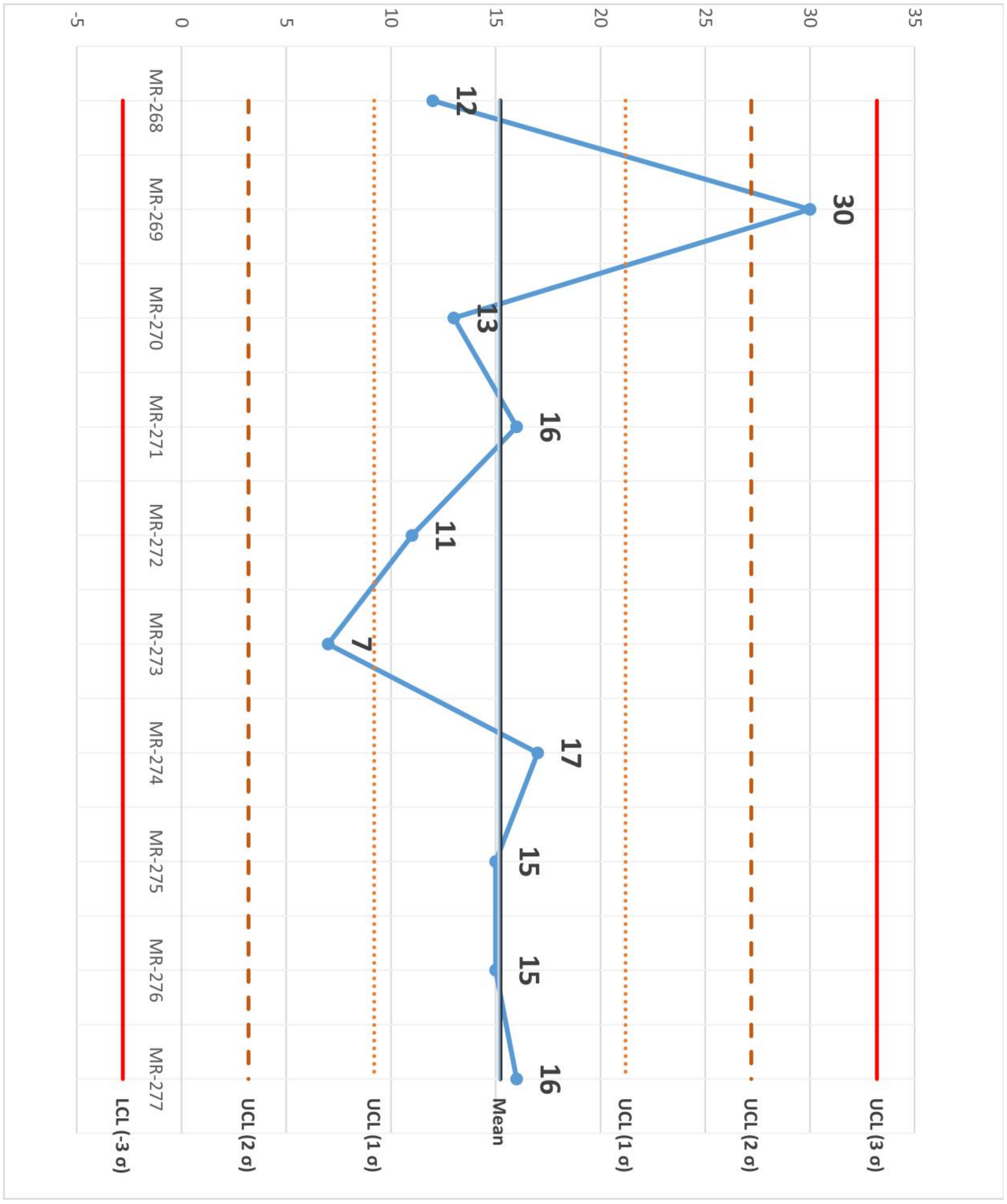


Table 6: Number of Failed Tickets through MR-277

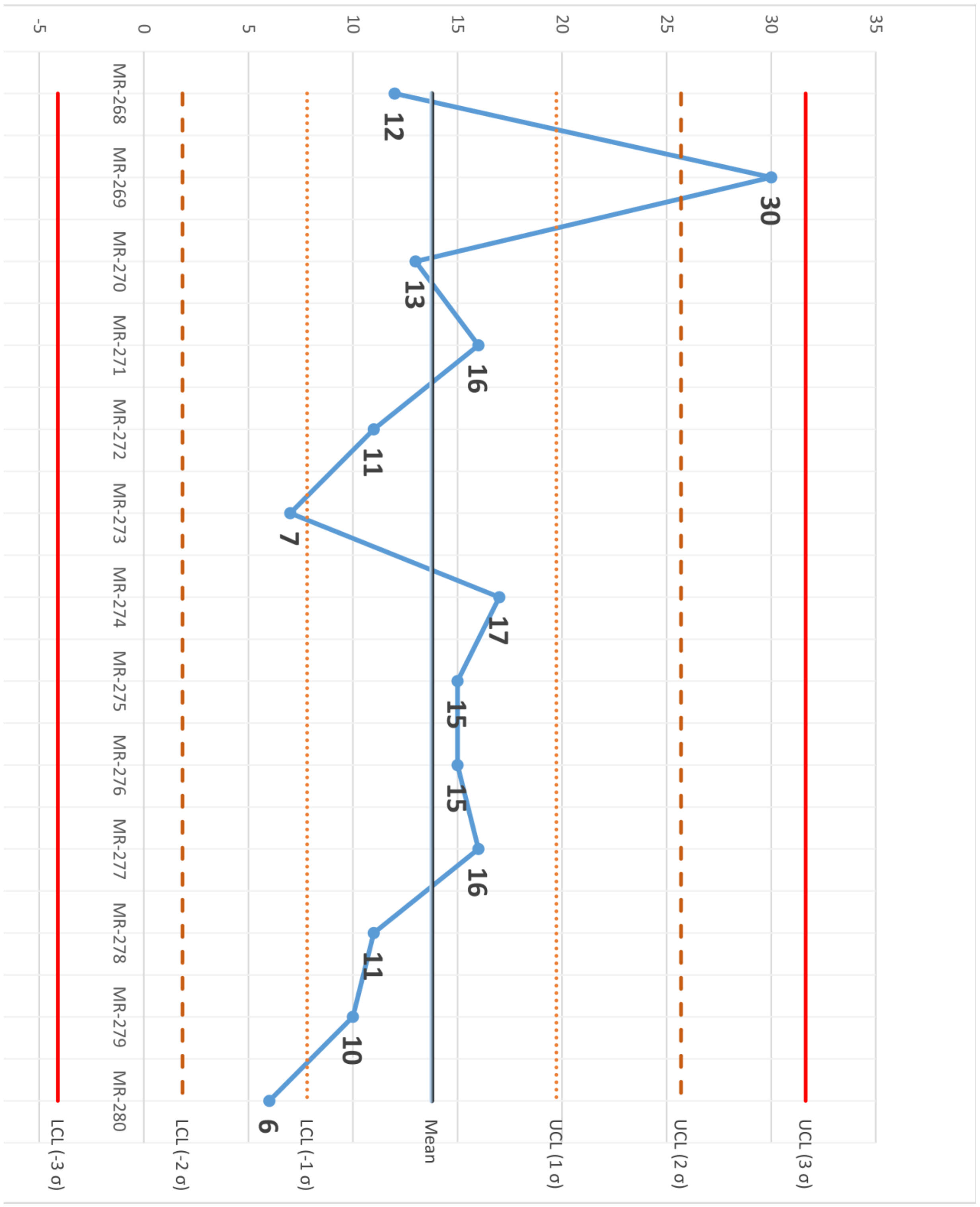


Table 7: Number of Failed Tickets through MR-280

Appendix 5: Number of Failed Incidents

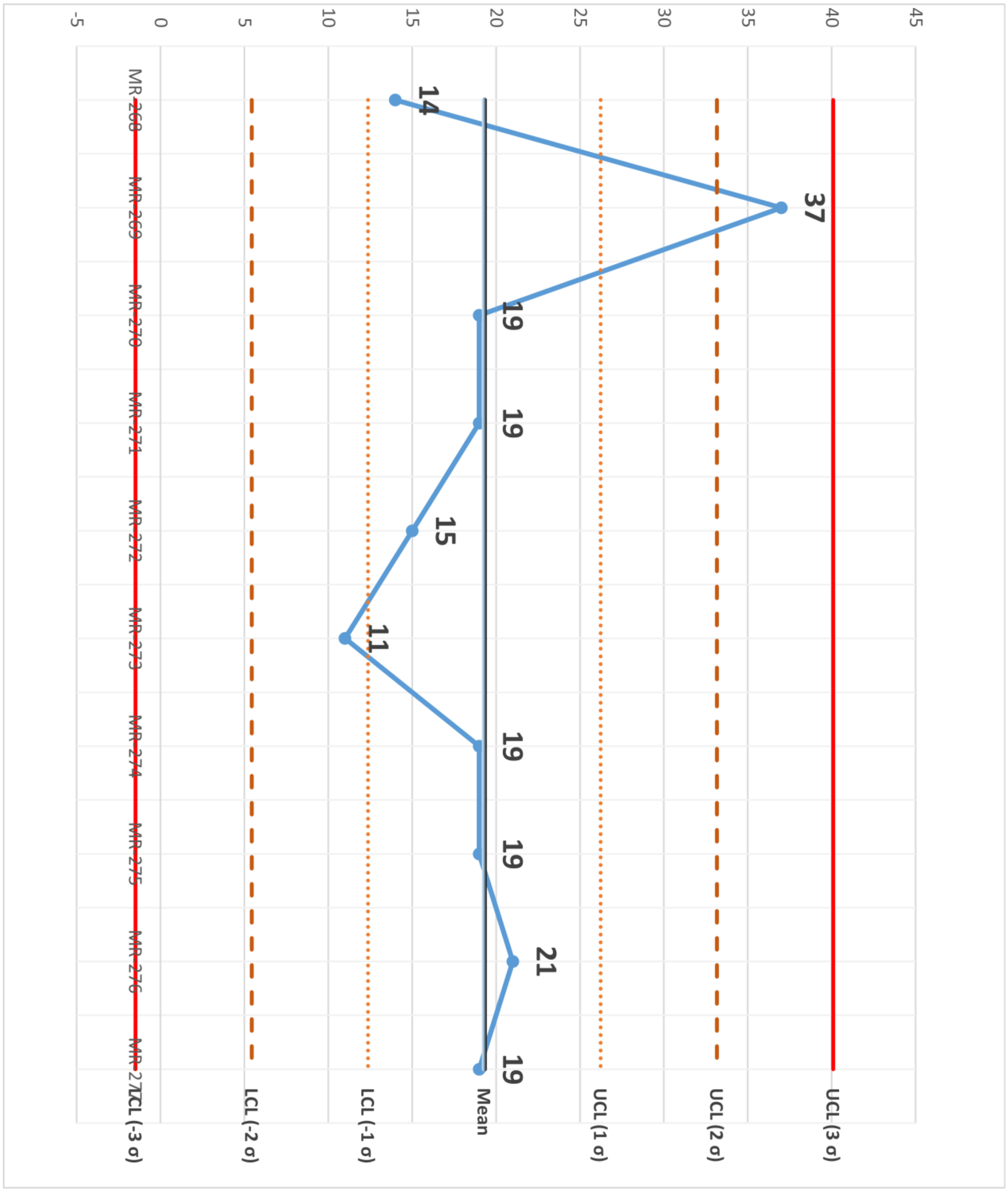


Table 8: Number of Failed Incidents through MR-277

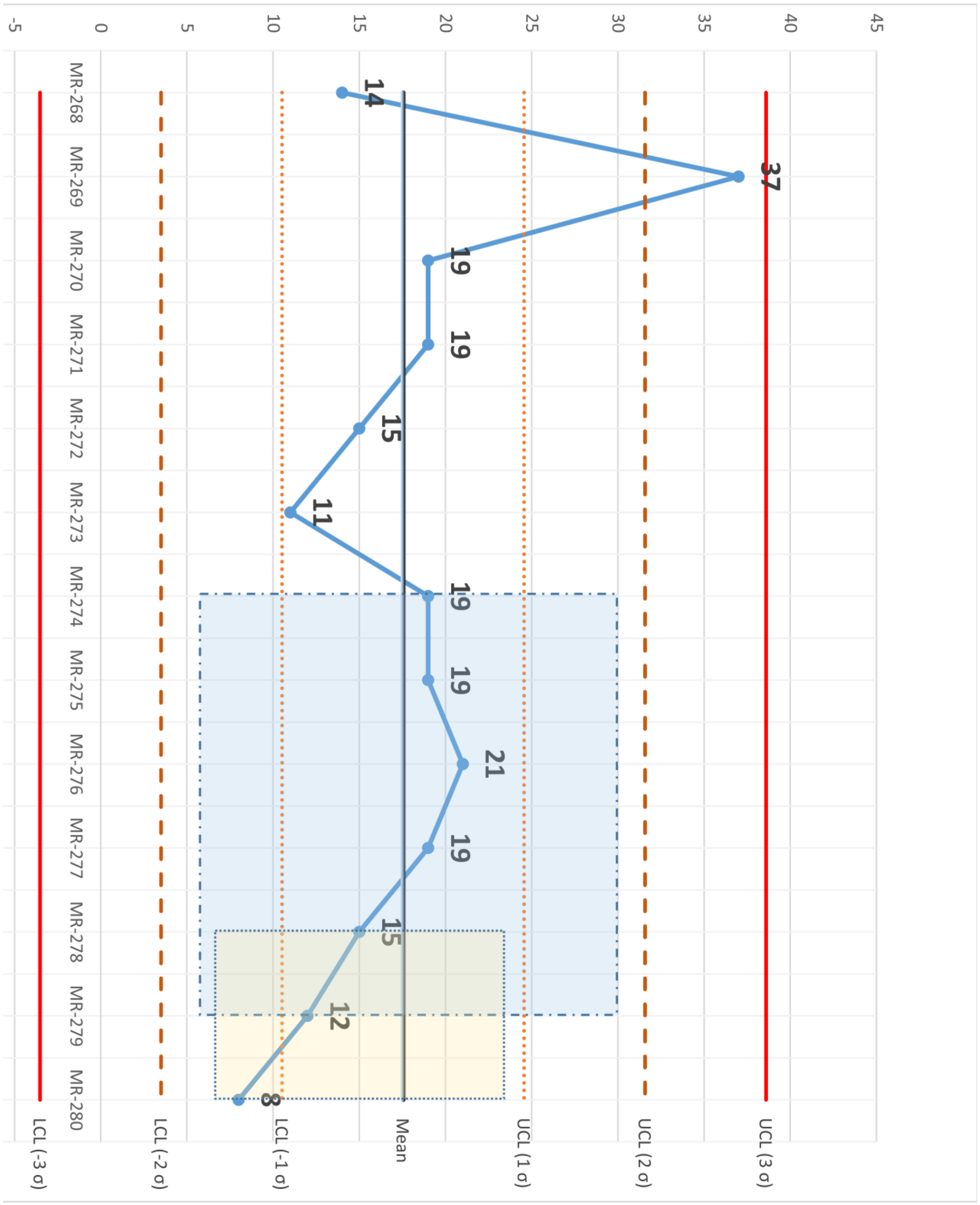


Table 9: Number of Failed Incidents through MR-280

The blue shaded area in the previous chart (Table 7) represents the period of time where SCDMV was maintaining two different COBOL systems. The yellow is the period of time where test plans were required with the tickets. Green is the overlap between the two periods of time.

Appendix 6: Number of Tickets per Release

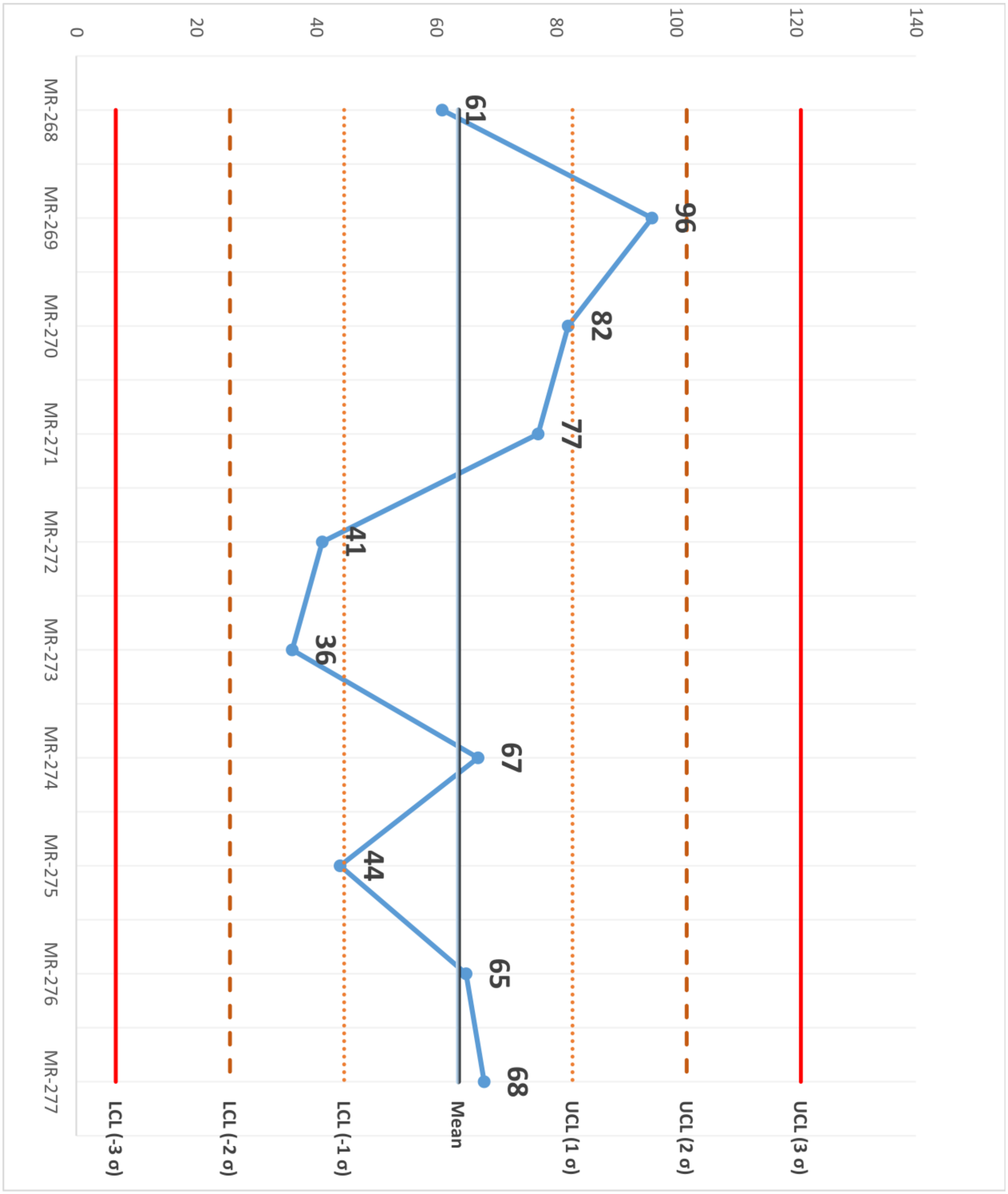


Table 10: Number of Tickets in Release through MR-277

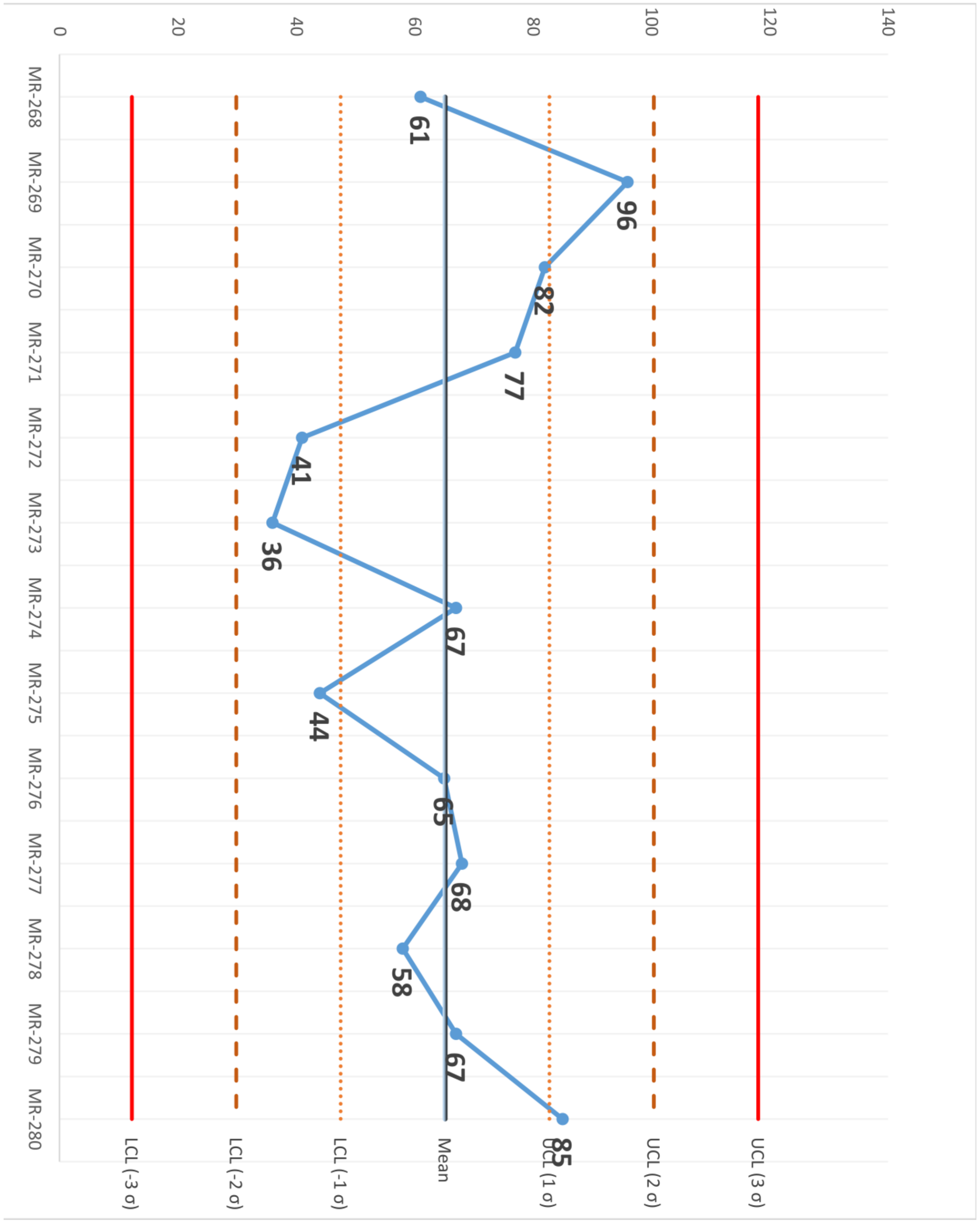


Table 11: Number of Tickets in Release through MR-280

Appendix 7: Number of Tickets Failed More than Once

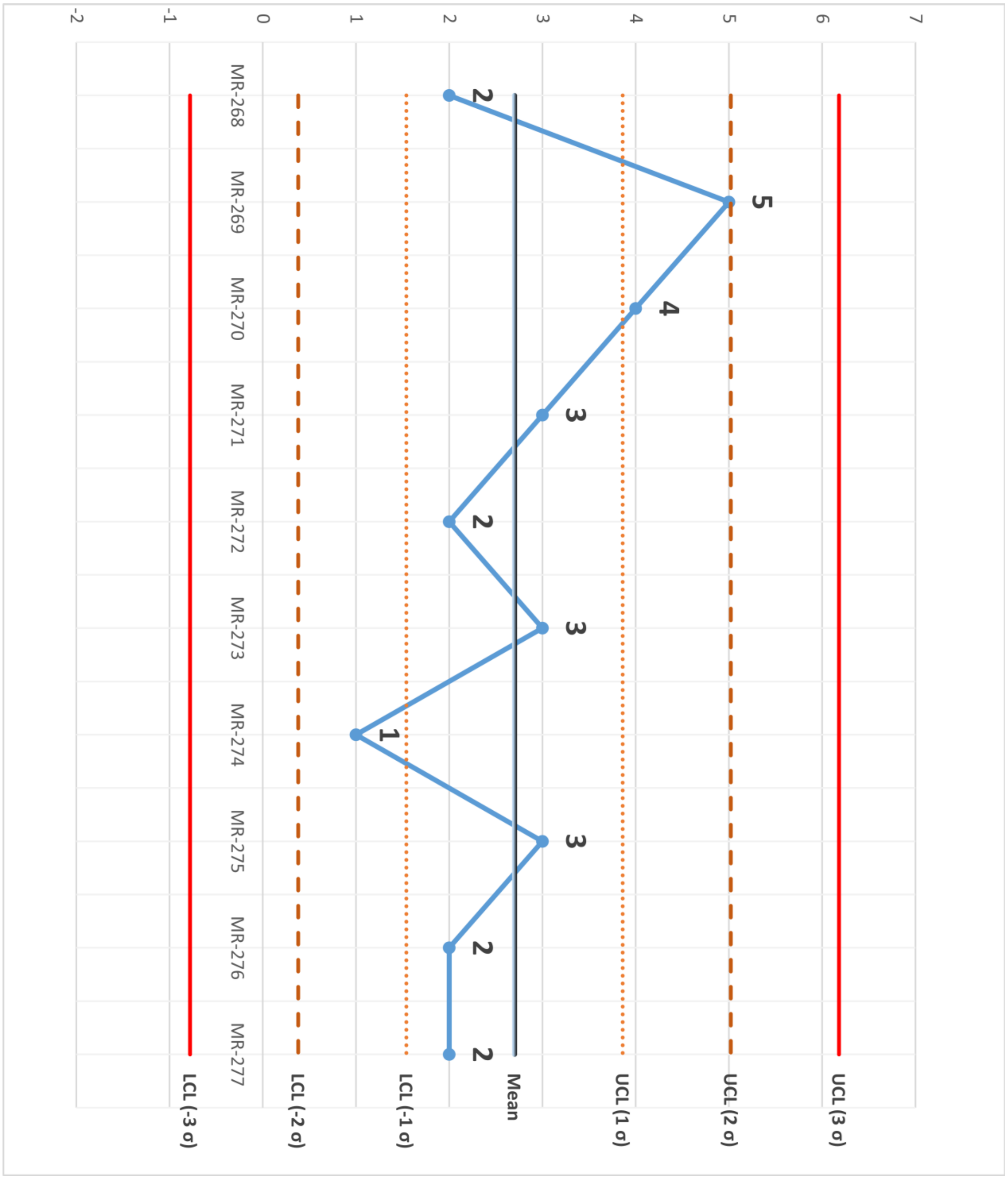


Table 12: Number of Tickets Failed More than Once through MR-277

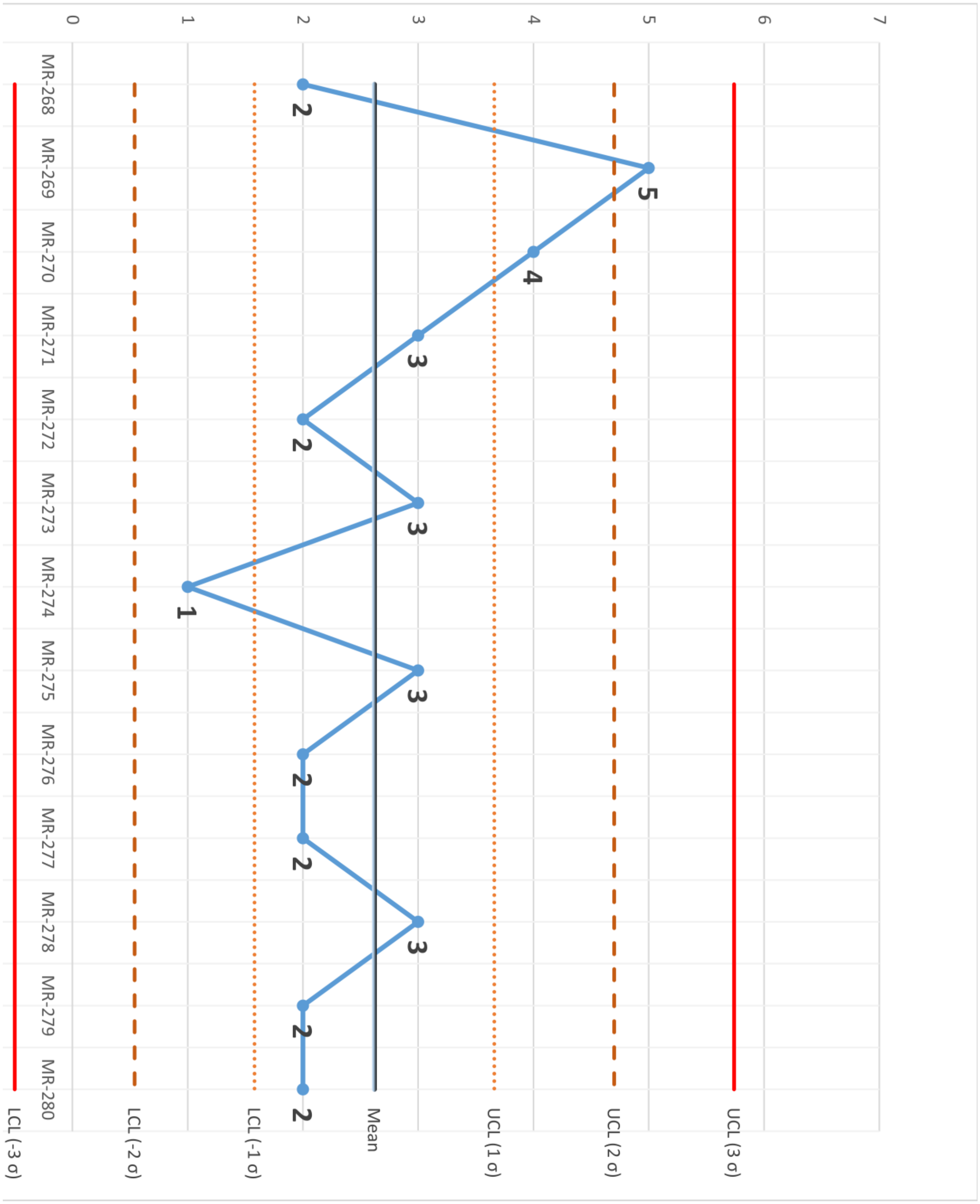


Table 13: Number of Tickets Failed More than Once through MR-280

Appendix 8: Percent of Failed Tickets

The percentage is calculated by dividing the number of failed tickets by the number of tickets in the release.

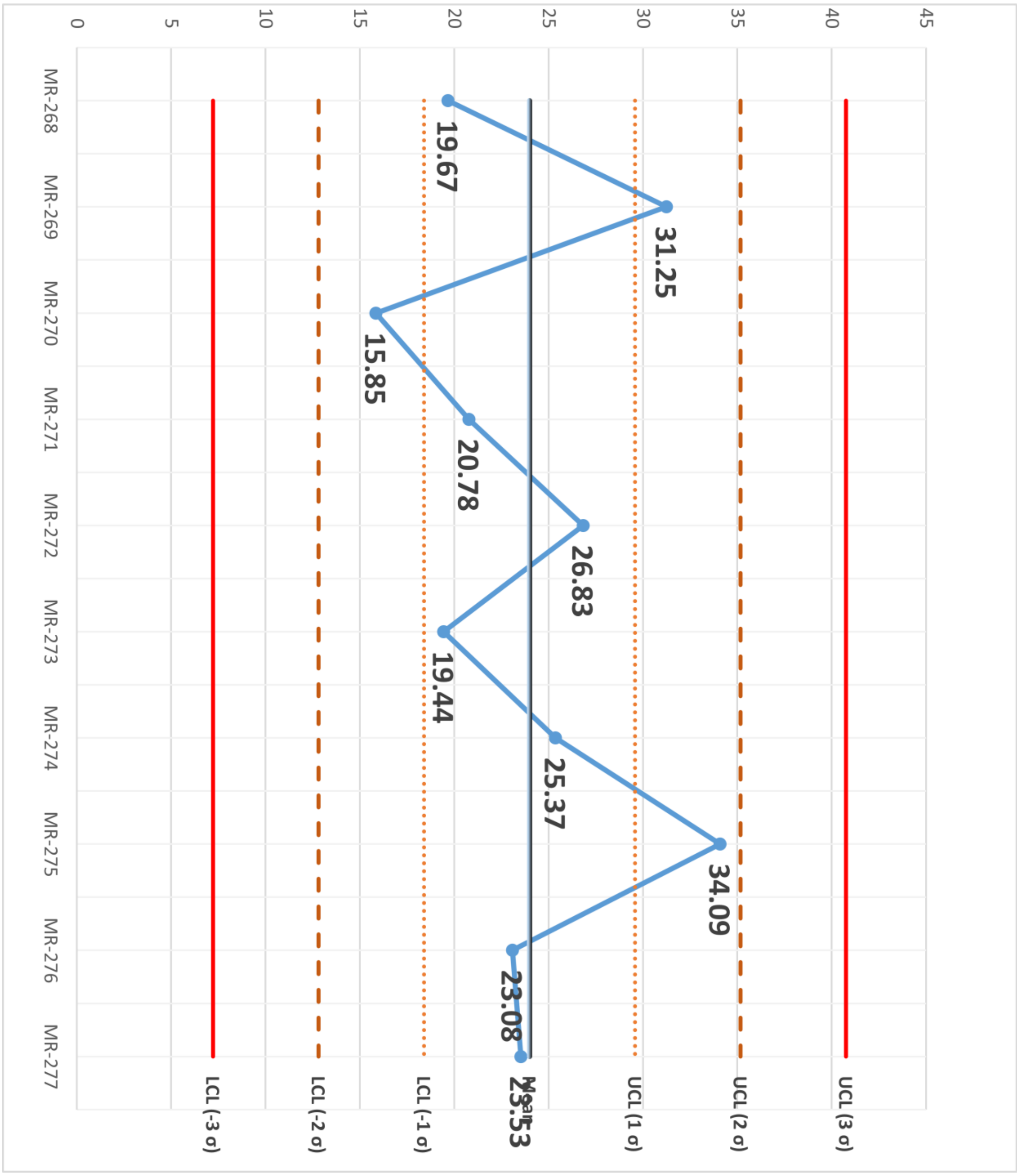


Table 14: Percentage of Failed Tickets through MR-277

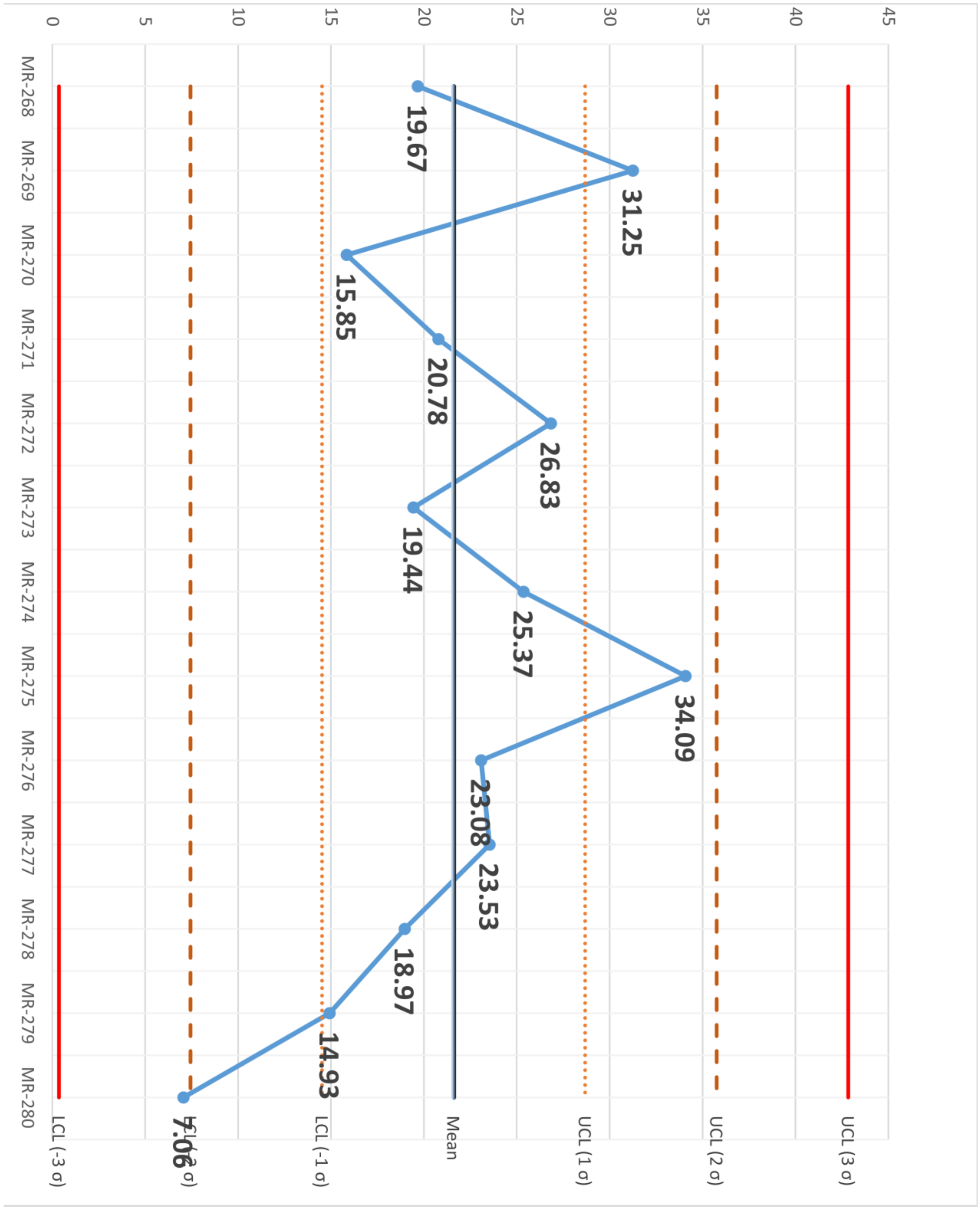


Table 15: Percentage of Failed Tickets through MR-280

Appendix 9: Percent of Failed Incidents

The percentage is calculated by dividing the number of failed incidents by the number of tickets in the release.

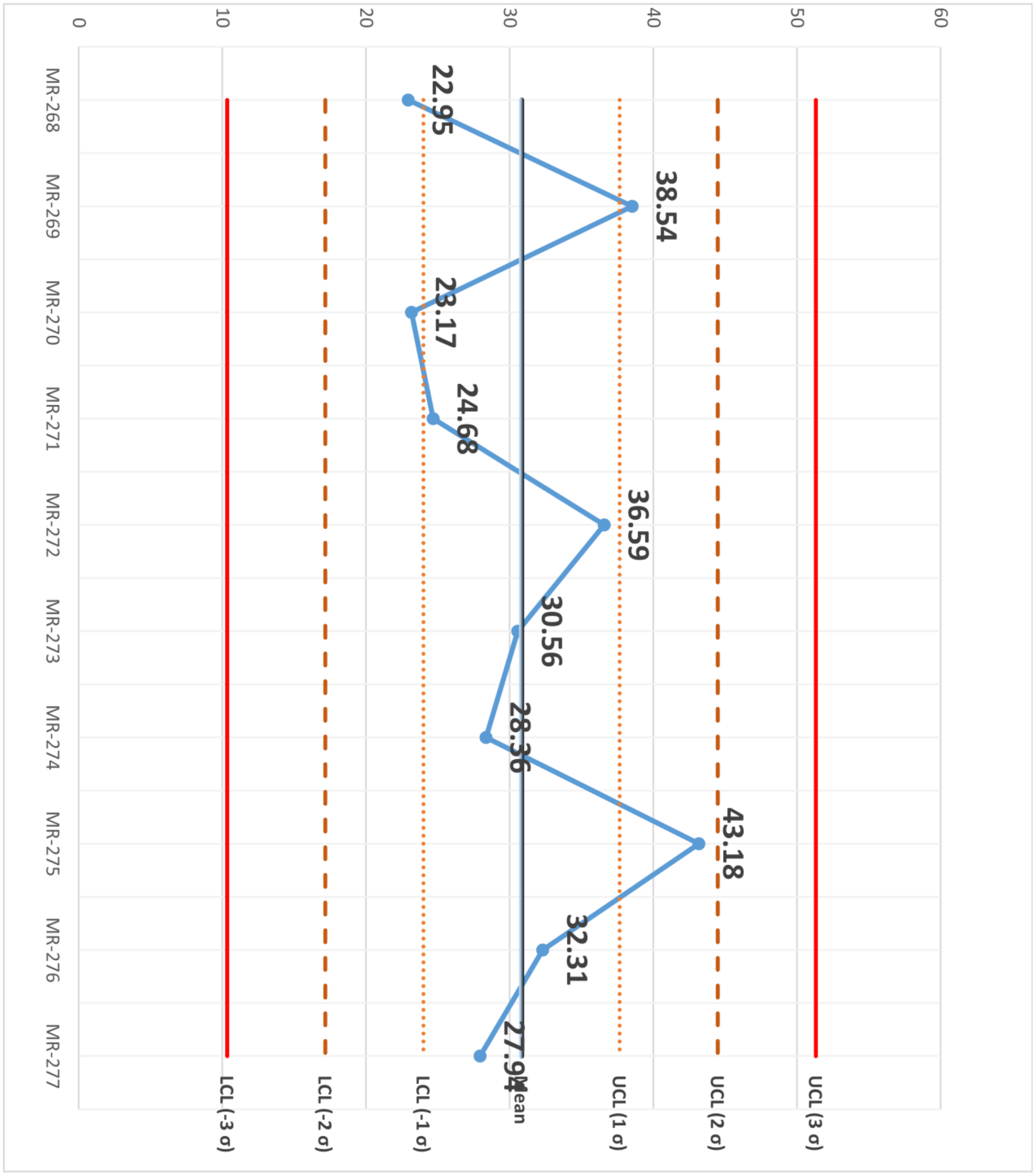


Table 16: Percent of Failed Incidents through MR-277

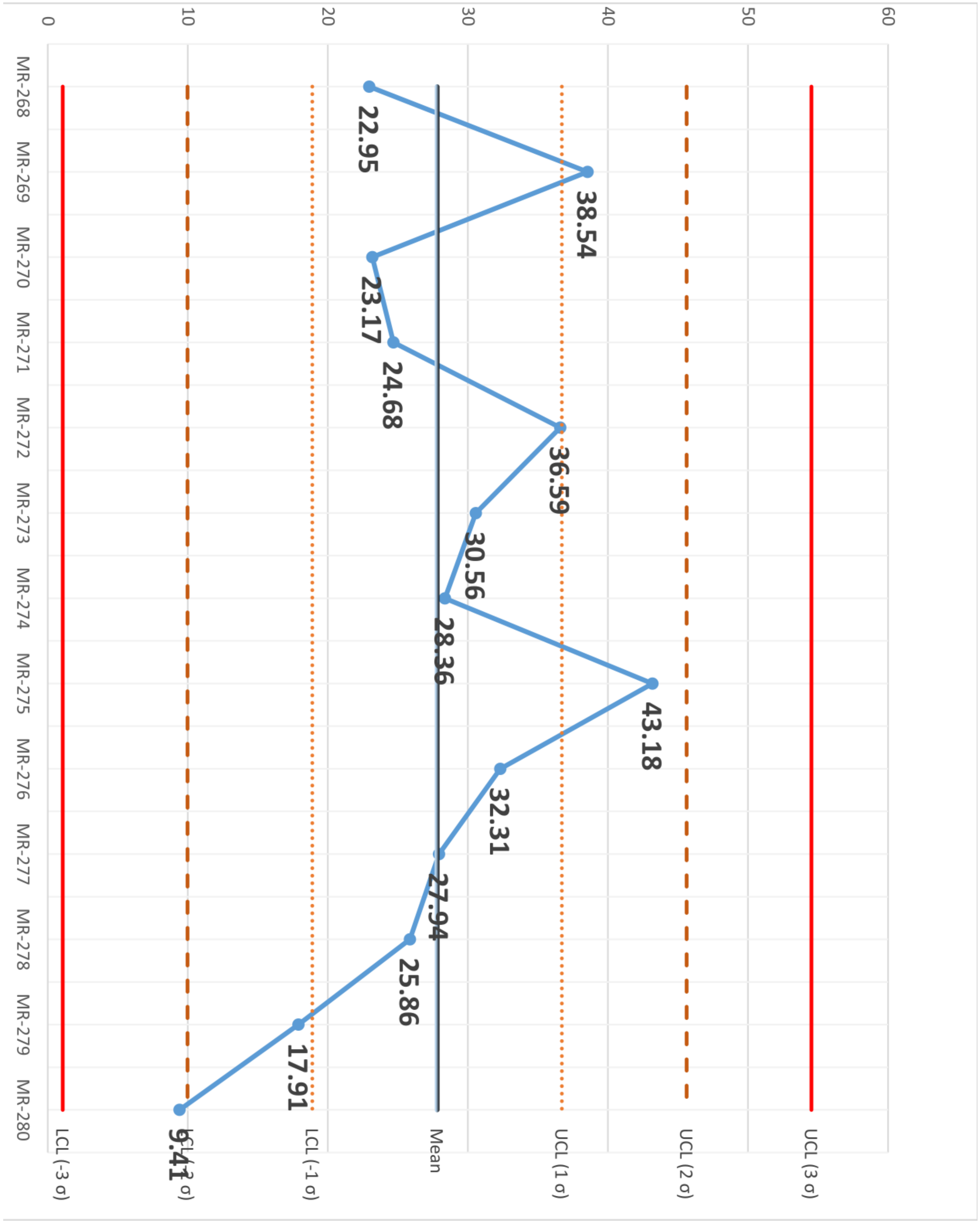


Table 17: Percent of Failed Incidents through MR-280

Appendix 10: Percent of Times Tickets Failed More Than Once

The percentage is calculated by dividing the number of tickets that failed more than once by the number of tickets in the release.

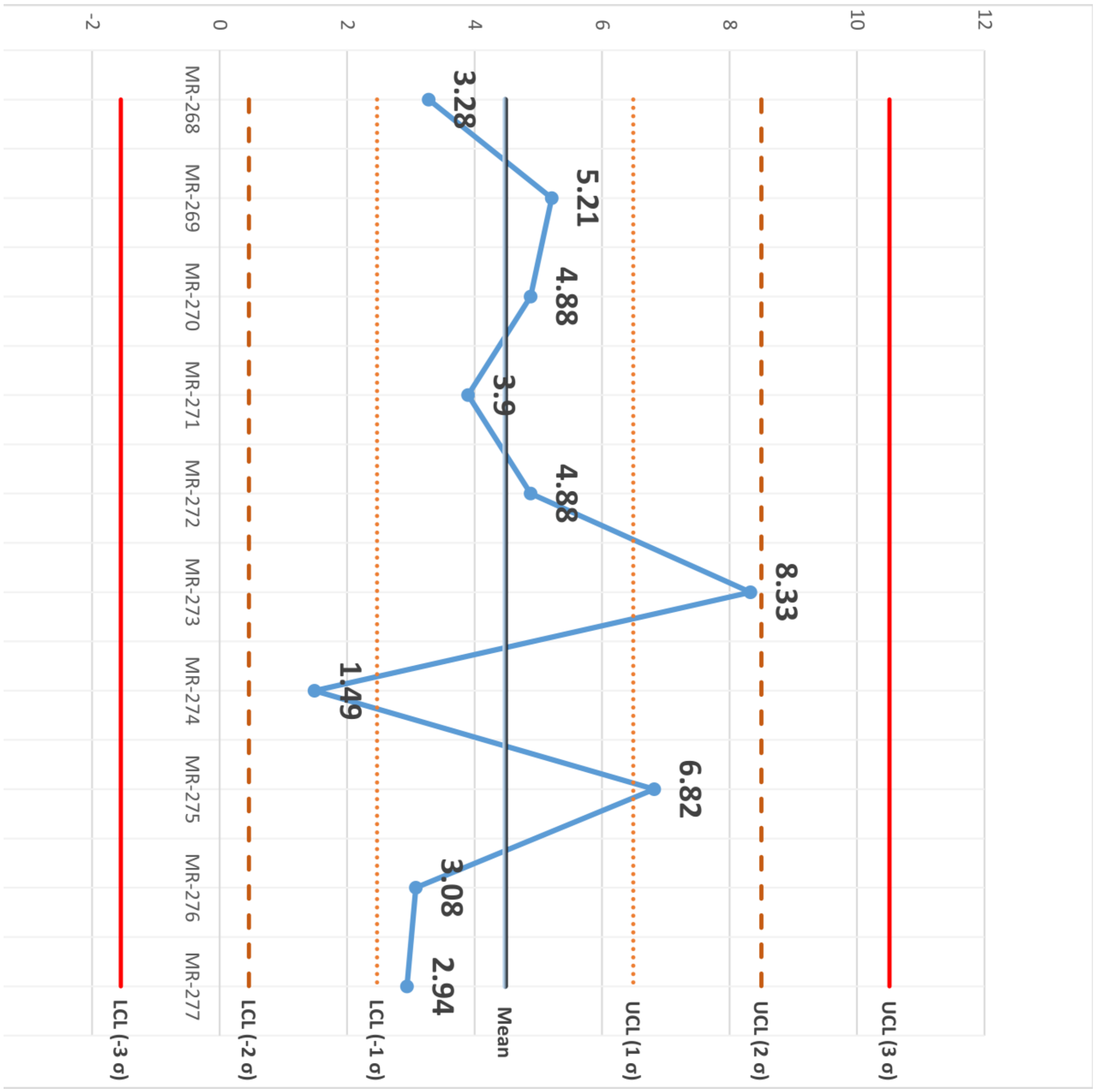


Table 18: Percent of Tickets that Failed More than Once through MR-277

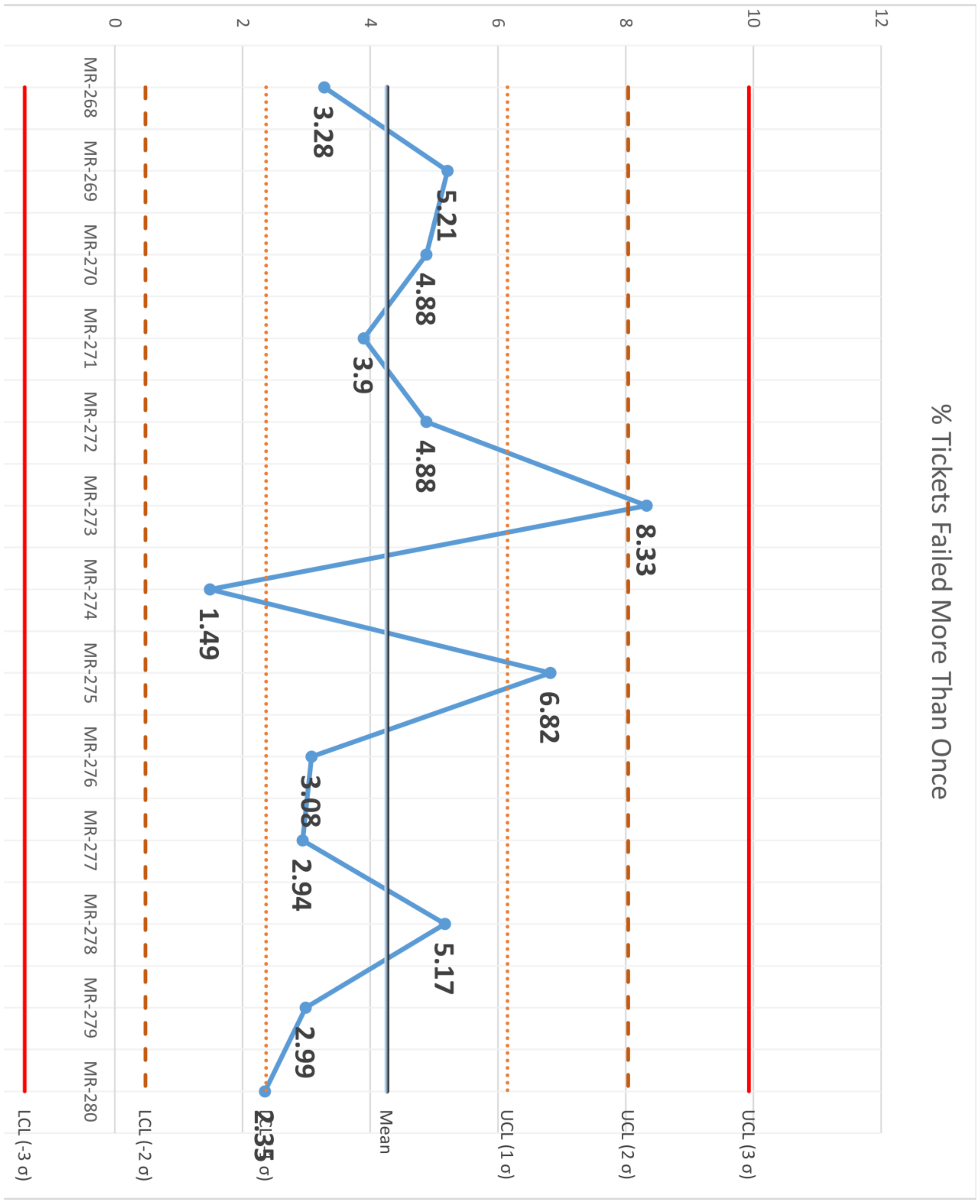


Table 19: Percent of Tickets that Failed More than Once through MR-280

Appendix 11: Failed Tickets vs Failed Incidents

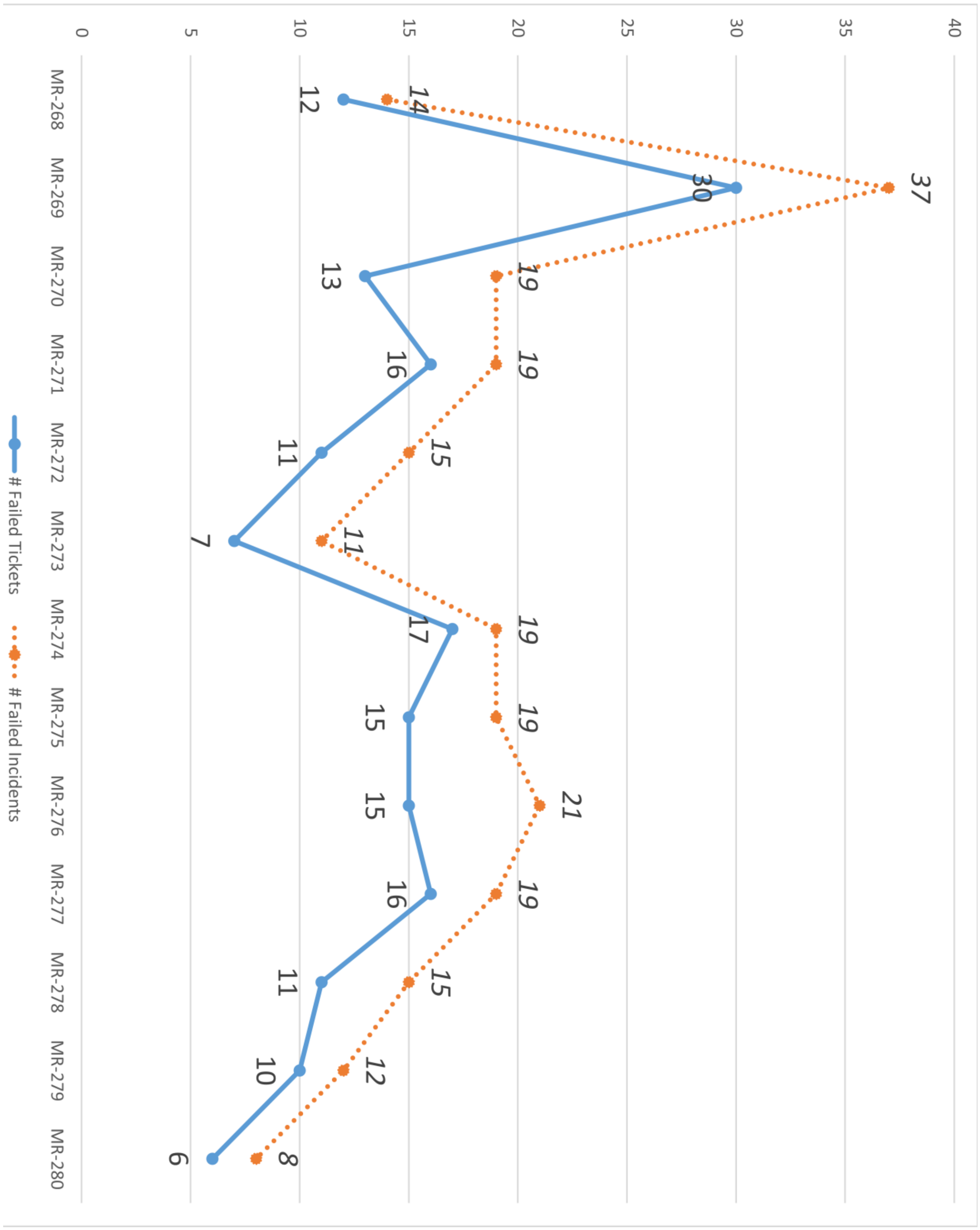


Table 20: Failed Tickets versus Failed Incidents through MR-280

Appendix 12: Incidents vs. Tickets Compared to Tickets in a Release

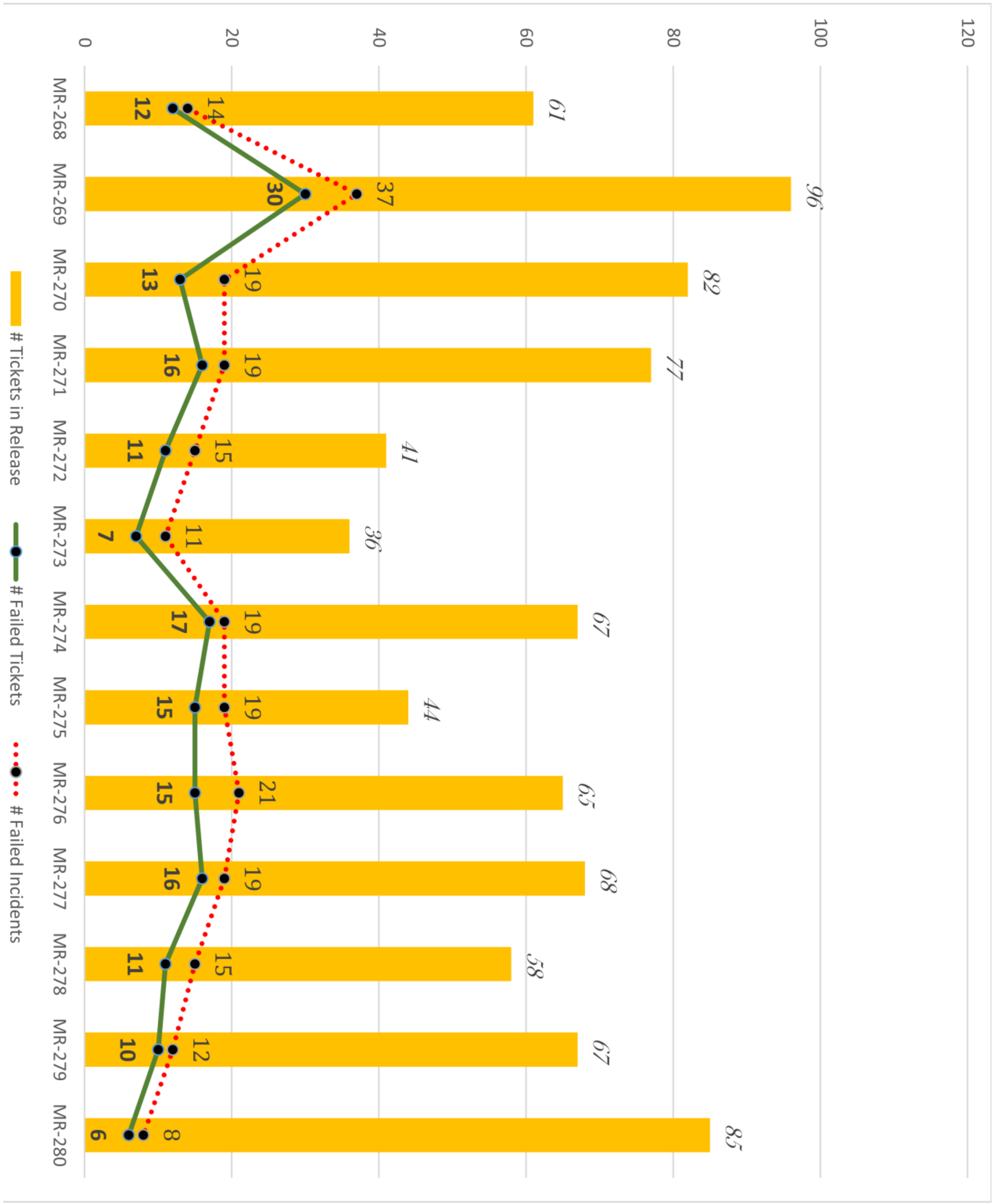


Table 21: Failed Incidents vs Failed Tickets Compared to Tickets in a Release through MR-280

Appendix 13: Survey

Most surveys from my experience have a return of ten to twenty percent. This survey had a return of 24%.

The questions

1. The software development, testing, and distribution process currently used is good because…

2. The software development, testing, and distribution process can be improved by…

The responses

The responses have not been edited in any way. The number refers to the response in order it was received. The responder's name is listed only if permission was given by them to use their name.

Question 1

1. For the most part, it provides a fairly reliable way to track tickets through the development process to production. The end result has been a basically good product in Phoenix.
2. Response
   - 2.1. The end product we produce is very stable and dependable. This is the ultimate goal.
   - 2.2. The current TFS process is the best the department has ever had. Very pleased with how this is implemented.
3. No response given
4. No response given
5. Response
   - 5.1. Software development process is enhanced by the requirement that specifications be written for enhancements. These details along with a business process model where the business analysts work hand in hand with the development staff to analyze the new or revisions to the systems, design and implement the modifications or new development make for a successful implementation.
   - 5.2. Testing in two environments, TEST and PREP, give business analyst the opportunity to test the fixes or enhancements for the code cycle focusing only on the changes to the system. Prep testing allows overall testing to ensure that no are no unexpected consequences to the fix or enhancement on the system. This is a sound method for testing.

   5.3. The distribution process has gone through many modifications based on technology enhancements and sometimes personal preferences. The distribution system used currently is good; however, we do have situations where unexpected code differences are introduced. The volume of code moved and the variety of modifications needed for a code move can contribute to code being overlooked. These situations are at a minimum.
6. It generally ensures that a stable product. In the long run, this saves the organization time and money.
7. The process is good because it's sequential and straight forward. At the end of it, it seems like applications have been well tested and are usually pretty stable after deployment.
8. Everything seems to be tracked and controlled well. We don't seem to have multiple developers working on the same programs at the same time as we used to.
9. It has made the testing better and speeded up the process when starting the testing.
10. William D. Neiswonger
    10.1. We have defined separation of duties. Developers may help with the compilation and testing of code, but the final decision to promote to production is not in the hands of the person that developed the change.
    10.2. Business analysts are used to help translate from business need to IT possibility. Almost like being an interpreter… From English to Geek! This is definitely a learned skill and our BAs are really doing a good job!
    10.3. As much as we complain about Service Manager, the tool is giving DMV a well-defined set of steps to follow. We have two main issues with this tool. One, we assume the tool should be able to handle every possible combination of requirements (it can't!). Two, all involved are fighting against the tool instead of trying to use it as it has evolved. Granted there are some things that the tool just does not do a good job with!
11. Anne Morgan
    11.1. It allows programmers and business analysts to focus on what the priorities of the tickets are.
12. It requires the involvement of the Business and IT. The business defines the requirements before IT develops a solution. When a solution is ready, the business must test and approve the results before the enhanced code can be move to Prep and ultimately production.

Question 2

1. A number of issues outside of our control has complicated the ability to adhere to our process, shortened the time frames, and has added a good bit of complexity to Phoenix and its support processes. However, someone I worked previously summed it up, "Results are King".
2. Response
    2.1. Allowing more tickets in a [Module Release] MR.
    2.2. Allow a new ticket to be considered in next MR instead of 2 or 3 MRs ahead.
    2.3. Testing should utilize the power of testing tools where possible.
    2.4. Code moves to production should not result in any downtime for Member Services. Law enforcement safety is at stake.
3. Automatic testing.

4. Micro Focus unified functional automated testing.
5. Response
   - 5.1. The processes can be improved by have clearer understanding of the components required for a move and the specific environments affected by modifications.
   - 5.2. For testing purposes, Business Analysts need to know when a program may impact areas other than where a fix or enhancement is needed. Examples are customer, we all aware that a customer change many impact any menu item in Phoenix, so testing of all customer entry points are necessary. Other changes in code are not so obvious and if developers could identify where the code change could impact other areas, it would assist in the testing process.
   - 5.3. Each environment should be described in details identifying the server, connection points to the server, etc.
6. Implementing continuous integration, automated testing, pair programming and peer reviews – all of which – help to ensure more maintainable, supportable, reliable and extensible software systems.
7. It may or may not be possible, but I think the process could be improved by having all the functionality we need in one system and eliminating the need to go back and forth between Service Manager, TFS, Build Forms, etc.
8. Response
   - 8.1. When a developer doesn't complete a build form/process properly. It falls on the BA to follow up with the developer to identify why the change hasn't been processed for retest. This often happens when the BA is waiting for a fix for a failed ticket.
   - 8.2. Also, it would be nice to have dedicated testers from each processing area. Vehicle Services is so short staffed that the processing areas cannot provide testing support during regression testing phase. We need testers that are subject matter experts from their area to fully test vehicle services.
9. I know there is always room for improvement but this process has come a long way and the improvement now are great.
10. William D. Neiswonger
    - 10.1. When a developer fails to meet the freeze date, moving code from one MR to the next presents challenges that we have not come up with a process to aid us.
    - 10.2. Developing code in multiple MRs at the same time is problematic because MR + 1 may not include code from MR. And it won't include the code from the current MR until it goes to production and is merged into main. Most of our coding effort happens in less than 10% of our code base. The odds of having the same program in subsequent MRs is almost guaranteed.
    - 10.3. I know there must be a reasonable limit to the tickets that developers can code and BAs can test. I don't know if there is a way to assign tickets a point value for complexity of coding and/or testing. We could process a bunch of low point tickets, where only a few high point tickets should be included in a push. And we should also consider functional area in the calculations.

10.4. Need 2 person verification that code was compiled correctly and TFS merges were completed properly. Person 1 does the work as designed by our current process. Person 2 is responsible for comparing TFS to the compile area, making sure the test area has the latest compiled versions. Merges in TFS should also include a verification step. It is super easy to forget a step or miss where something didn't get copied because it was in use or failed to compile successfully. Things happen!

11. Anne Morgan

11.1. Allowing BAs and programmers to determine what goes in each build. "This build is frozen because we already have x number of tickets" doesn't make sense. If a programmer can code a fix, and a BA can test it, the ticket should be included.

11.2. With all the merging between builds, it is far too difficult to see the history of changes for a program. It would be good to have a version that just lists the changes. That is programmers check in comments only in a list with no indenting. Removing the "merged by" comments in between would be very helpful.

11.3. Ensuring the merge process is complete and correct before a build goes to test.

11.4. Approving Technical Specifications in a more timely manner. They should be approved within 24 hours. Once the Tech Spec is approved, the technical reviewer should not require changes if the spec is adhered to.

11.5. Automating the process when PRs are moved from one MR to another. With the purported merging ability of TFS, programmers should not have to repeat their work when tickets change MRs. CM or TFS administrators should be able to use the TFS functionality and move tickets from one MR to another when required without intervention from the programmers. Especially when there are no other code changes on top of the one being moved, which was the case here.

11.6. Always having two working branches in TFS. The branch that is current, plus the next branch. When one MR goes into PROD, MR+2 branch should be opened.

12. The software development, testing, and distribution process can be improved by more planning and communication. It appears that some programming enhancements are not delivered to the business in time to perform through testing. When the business and IT decide what task will be included in the next release, they should assign a business analyst, programmer and programmer manager. It should be the responsibility of the programmer manger to provide a status reports to show the task are assigned and are scheduled to be delivered in a timely manner so testing can be completed

Summarization of Survey

Good (Listed alphabetically)	*Count*
Business Analysts	1
Distribution	2
No Response	2
Reliable	5
Separation of Duties	2

Specifications	1
Testing	2
TFS and Service Manager	2

Improvement Suggestion (Listed alphabetically)	***Count***
Approving Technical Specifications in a timely manner	1
Combing Tools (Service Manager/BuildForms/TFS)	1
Complete buildform	1
Dedicated Testers	1
Document Environments/Servers	1
Increase number of tickets in a release	2
No downtime during code promotion	1
No suggestion	1
Planning/Assignment	2
Reduce Complexity	4
Testing (Automated) Tools	4
TFS Merge	1

Appendix 14: Causes of Failed Tickets

Assigned Grouping

Each response for a failed ticket was reviewed and placed into a grouping based on the response from the Developer and BA.

1 = Developer did not follow test plan
2 = Developer Error
3 = Misunderstood ticket
4 = Did not Respond
5 = Scope Creep
6 = Test Plan not complete

MR-277

Ticket	Response	Grouping
RR289388	Found something else	5
RR313552	Developer misunderstood the requirements	3
RR313692	*No response*	4
RR314979	Did not understand existing design	3
RR316660	Programmer error	2
RR316683	Specs not clear	2
RR316751	Specs not clear	5
RR317030	Developer missed	3
RR317292	Developer did not promote	2
RR319109[20]	Developer did not have tools (printer) to test	1
RR319109	Developer test did not go far enough	1
RR320778	Code added to fix the stated problem inadvertently affected other functionality.	5
RR321591	Developer did not know enough about how to test	1
RR322117[21]	Developer could not test	1
RR322117	*No response*	4
RR322117	Specs not clear	5
RR323716	Missed in coding	5
RR329532	Missed in coding	5

Table 22 - Causes of Failed Tickets MR-277

[20] RR319109 was failed more than once; therefore, the same questions were asked each time it failed.
[21] RR322117 ibid.

MR-278

Ticket	Response	Grouping
RR315399[22]	Test plan did not include all steps	6
RR315399	Developer did not follow test instructions	1
RR318839	Ticket impacted other areas that were not thought of	5
RR322936	Developer did not follow test instructions	1
RR324070[23]	Developer only coded in one area	2
RR324070	Business spec not complete	5
RR324658[24]	Scope creep	5
RR324658	Misunderstanding of the ticket	3
RR324658	Scope creep	5
RR325204	Ticket impacted other areas that were not thought of	5
RR326986	Developer did not follow test instructions	1
RR328547	Test plan did not include all steps	6
RR329284	Developer did not follow test instructions	1
RR330232	Exposed another error	5
RR331684	*No response*	4

Table 23 - Causes of Failed Tickets MR-278

MR-279

Ticket	Response	Grouping
RR323880	Scope creep	5
RR331246	Issue found while testing	5
RR331520	Scope creep	5
RR334066	Managerial. Ticket not completely tested	1
RR335802	Did not know all impacted parts	5
RR336233[25]	Developer did not follow test instructions	1
RR336233	Developer did not follow test instructions	1
RR336579	Original code not available	2
RR337478[26]	Large ticket. Lots of moving parts.	3
RR337478	Large ticket. Lots of moving parts.	3
RR337844	Scope creep	5

Table 24 - Causes of Failed Tickets MR-279

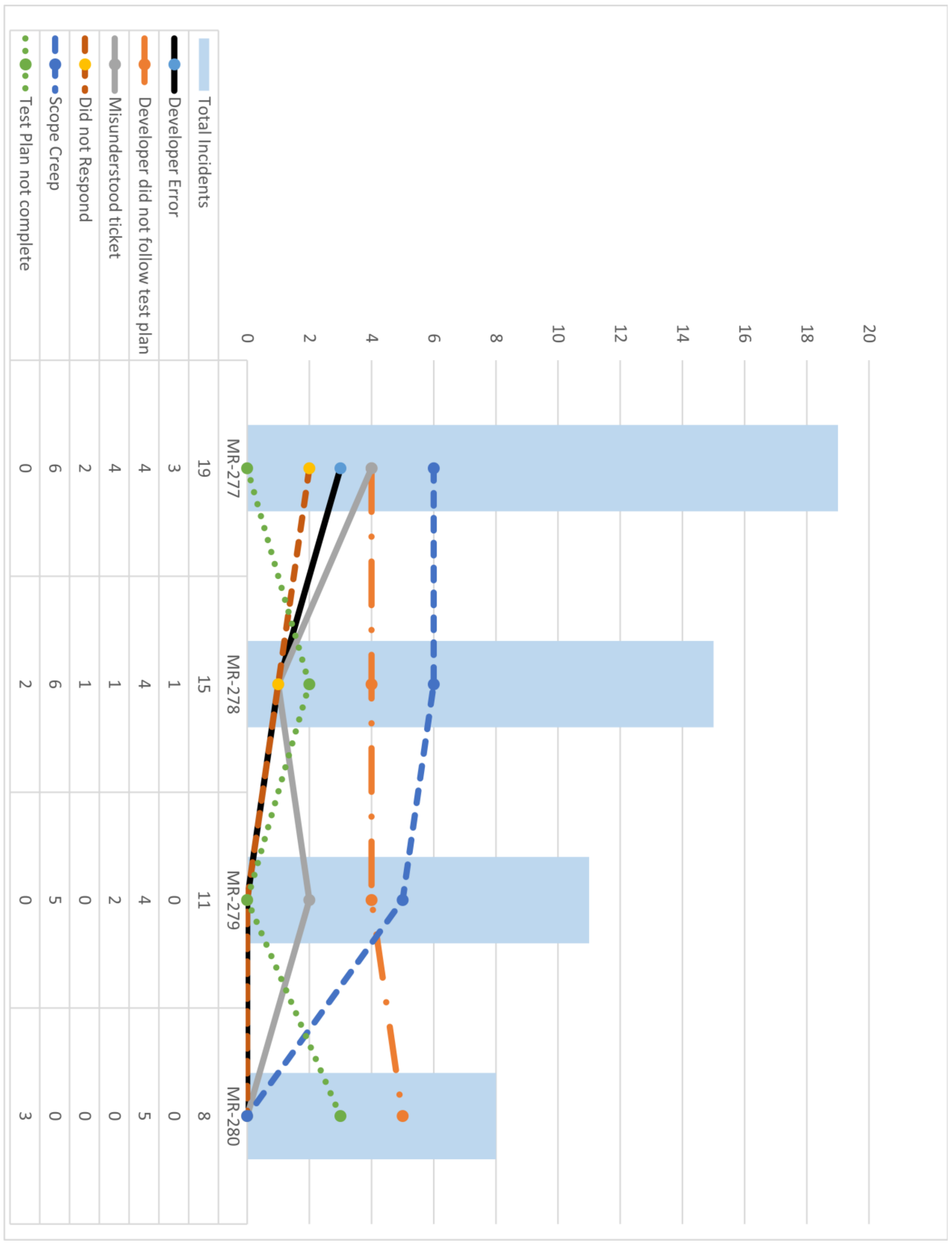
[22] RR315399 was failed more than once; therefore, the same questions were asked each time it failed.
[23] RR324070 ibid.
[24] RR324658 ibid.
[25] RR336233 ibid.
[26] RR337478 ibid.

MR-280

Ticket	Response	Grouping
RR335841	Unexpected data	6
RR338643	made a “cut and paste” mistake	1
RR338855[27]	Per developer request: Please fail so I can get correct DB mod in	1
RR338855	Code could only be tested in PREP – RE: external connect via PREP to TEST	1
RR341048	Out results are successful when we test using the Uni Tool, but there is a different result in the AAMVA tool	6
RR342348[28]	Small segment of ticket was not working. Fix was already submitted to rectify problem.	1
RR342348	Developer did not test CDL in Dev	1
RR344563	Out results are successful when we test using the Uni Tool, but there is a different result in the AAMVA tool	6

Table 25 - Causes of Failed Tickets MR-280

[27] RR338855 was failed more than once; therefore, the same questions were asked each time it failed.
[28] RR342348 ibid.



	MR-277	MR-278	MR-279	MR-280
Total Incidents	19	15	11	8
Developer Error	3	1	0	0
Developer did not follow test plan	4	4	4	5
Misunderstood ticket	4	1	2	0
Did not Respond	2	1	0	0
Scope Creep	6	6	5	0
Test Plan not complete	0	2	0	3

Table 26 - Grouping Summary

Appendix 15: Notification Email

Email to all Developers and BAs

As part of the CPM class, I have had to study a section of our processes to see where and possibly how we can make improvements. Working with Robert and based on responses from BAs and Developers when a failed ticket is initiated, I have zeroed in on the acceptance testing of tickets (or lack thereof). Therefore, beginning with MR-278, any ticket that is not already in a code submitted status will need to have an acceptance test plan attached.

The BA and Developer will collaborate on the plan prior to any coding taking place. That test plan will be added to the Documentation tab of the RR. After the coding is completed and before the Developer turns the ticket over to CM, the Developer will execute that plan. If the ticket passes the plan, the Developers will indicate on the Documentation tab that the acceptance test was executed and successful. After that CM will move the code to Test where the BA will execute the plan along with any other testing that the BA deems necessary.

If a BA or CM finds that code has been submitted after July 3rd, 2019 without a defined and executed acceptance test, they will return the ticket back to development.

This is a study to see if this makes a difference. Please let me know if you have any questions.

Endorsement from the Application Manager

I am very much in favor of this concept. We will perform a post implementation analysis after the next couple / three pushes to see how this effects our delivery quality.

I have been looking at the failed tickets as they have been reported and while this will not catch every issue, it's a start.

My experience has always been that the more "appropriate" testing that the programmer can perform before turning the changes over to the BA the smoother the entire process works.

This takes a conscious effort to understand the ramifications of changes that are being injected into our systems and requires significant collaboration between the analyst and the technician. This may also involve having the BA sit with the Programmer as they are performing their code testing.

There will be limitations to this concept. Obviously we don't have the same level of data in the Dev databases and sometimes setting up multiple conditions in that environment is not viable. However this practice should not pose an undue burden on our development process. At least no more than having a failed ticket due to lack of testing and putting it all the way back through the development process. If we can avoid this extra work, the better off we'll all be.

I would like to see if a side effect of this is the reduction of "fixes" to "problems" that have been caused by changes injected in to our systems.

Glossary

Word/Acronym/ Phrase	**Definition**
BA	Business Analyst
CM	Configuration Management
CSR	Customer Service Representative
Failed Incident	Each time that a ticket is failed. A ticket can be failed multiple times before it is resolved.
IT	Information Technology
IVP	Install Verification Plan
MR	Module Release
Regression	Re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change.
SCDMV	South Carolina Department of Motor Vehicles
SCSM	Microsoft System Center Service Manager. The ticketing system used by SCDMV.
SDLC	Software Development Life Cycle
System Test	A level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications.
TFS	Microsoft Team Foundation Server. The library management system used by SCDMV.